COPY OF PAPERS
ORIGINALLY FILED

#17
8 D.
9-19-02

Attorney Docket No. YO998-522

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Patent Application

Applicant(s):	L.D. Comerford et al.
Docket No.:	YO998-522
Serial No.:	09/460,913
Filing Date:	December 14, 1999
Group:	2654
Examiner:	Abul K. Azad
Title:	Methods and Apparatus for Contingent Transfer and Execution of Spoken Language Interfaces

---

AFFIDAVIT UNDER 37 C.F.R. §1.131

Assistant Commissioner for Patents
Washington, D.C. 20231

Sir:

We, the undersigned, hereby declare and state as follows:

1. We are the joint inventors of the invention described and claimed in the above-referenced U.S. patent application.

2. At least as early as February 1998, we conceived an invention falling within one or more of the claims of the above-referenced patent application.

3. At least prior to January 29, 1999, the invention falling within one or more of the claims of the above-referenced patent application was reduced to practice by implementing it in hardware and in software code.

4. At least prior to January 29, 1999, we prepared a three-page Invention Disclosure entitled "Disclosure YOR8-1998-0518." The Invention Disclosure describes our above-noted invention. A copy of the Invention Disclosure is attached hereto as Exhibit 1.

5. At least prior to January 29, 1999, we prepared a six-page Memorandum entitled "The Personal Speech Assistant Project." The Memorandum further describes our above-noted invention. A copy of the Memorandum is attached hereto as Exhibit 2.

6. The description in the Invention Disclosure and Memorandum documents evidence conception of an invention falling within one or more of the claims of the above-referenced patent application at least prior to January 29, 1999.

7. All statements made herein of our own knowledge are true, and all statements made on information and belief are believed to be true.

8. We understand that willful false statements and the like are punishable by fine or imprisonment, or both, under 18 U.S.C. §1001, and may jeopardize the validity of the application or any patent issuing thereon.

Date: 9/4/02

[illegible]

Liam D. Comerford

Date: 5 SEP 2002

[illegible]

David Carl Frank

OIPE JCE
SEP 11 2002
PATENT & TRADEMARK OFFICE

COPY OF PAPERS
ORIGINALLY FILED

Disclosure YOR8-1998-0518

Created By: Liam D Comerford
Last Modified By: Liam D Comerford

*** IBM Confidential ***

Required fields are marked with the asterisk (*) and must be filled in to complete the form .

Summary

Status	Submitted
Processing Location	YOR
Attorney/Patent Professional	Paul J Otterstedt/Watson/IBM
Submitted Date	
Owning Division	RES
PVT Score	61

Inventors with Lotus Notes ID's

Inventors: Liam D Comerford/Watson/IBM, David Frank/Watson/IBM

Inventor Name > denotes primary contact	Inventor Serial	Div/Dept	Manager Serial	Manager Name
> Comerford, Liam D	818655	22/904N	200736	Gopalakrishnan, Ponani S. (Gopa)
Frank, David C	887191	22/921N	111498	Shea, Dennis G.

Inventors without Lotus Notes ID's

IDT Selection

Main Idea

***Title of disclosure (in English)**

Contingent User Interface Specification Protocol

***Idea of disclosure**

1. Describe your invention, stating the problem solved (if appropriate), and indicating the advantages of using the invention.

Description:

User interface elements for speech oriented systems are described as text data including command vocabularies, prompts, scripts, hardware profiles, and grammars. These elements are cached for search, management and use by a "Supervisor" software component. The supervisor responds to notification of a new command target by searching the cache for the target identifier, and failing to find the identifier queries the target for its ui element set. On completion of the protocol, data required to initialize ui engines is available as is data required to shape the responses of the user interface. Cache management includes an LRU method and data defragmentation in order to make best possible use of the limited memory in embedded devices.

Problem:

In the normal operation of a speech enabled application, the set of vocabulary words, spoken prompts, and other user interface components which are necessary and sufficient for the operation of the application will change. Further, as the user moves between applications or addresses speech aware devices other than a computer, this necessary and sufficient set of UI components will also change. If the active set of these components is not kept in synchronization with the target of the user's active address, the interface behavior will be useless, annoying and perhaps bizarre and destructive.

Advantage:

Speech UI elements are kept synchronized through a UI element cache management protocol. This keeps the working recognition vocabulary up to date and at a minimum size, thus maximizing recognition accuracy. Similarly, the response time for issuing prompts or acting on commands is minimized.

2. How does the invention solve the problem or achieve an advantage,(a description of "the invention", including figures inline as appropriate)?
see description above

3. If the same advantage or problem has been identified by others (inside/outside IBM), how have those others solved it and does your solution differ and why is it better?

UI element problems are most thoroughly explored in GUI environments such as Windows '95. These solutions are focused on making engines into generically accessed components rather than dynamically managing the speech UI.

4. If the invention is implemented in a product or prototype, include technical details, purpose, disclosure details to others and the date of that implementation.
The invention has been implemented in rudimentary form in the Personal Speech Assistant Emulator.

***Critical Questions (Questions 1 - 7 must be answered)**
Patent Value Tool (Optional - this may be used by the inventor and attorney to assist with the evalu
Post Disclosure Text & Drawings

(Form Revised 12/17/97)

Disclosure YOR8-1998-0518

Contingent User Interface Specification Protocol

Prototype claims:

1. In a Personal Speech Assistant, the method of creating new Spoken Language Interface Files by the steps of:

> determining that an event has no corresponding data in the Spoken Language Interface Data Structure.
>
> broadcasting a request for SLI data to the application platform originating that event
>
> storing the returned SLI data
>
> de-referencing the event entry in order to implement the interface.

2. The method of claim one in which the application platform is an appliance.

3. The method of claim one in which the application platform is an information system.

4. The method of claim three in which the information system is a location specific service.

OIPE
SEP 11 2002
PATENT & TRADEMARK OFFICE

The Personal Speech Assistant Project

Liam Comerford, David Frank, Dominico Manildo, Steve Mastrianni, Jan Sedivy

Introduction

Human beings use two characteristic means for accomplishing work in the world. They collaborate with other people through language (typically in the form of speech and listening) and they operate on the material world with tools.

Collaborative relationships among people span a range of forms from fully equal partnerships (in which the skills and observations of each party complements the other), to entirely directive relationships in which one party directs and the other(s) simply perform according to instruction.

As the technology of our tool culture progresses, we find ourselves on the verge of creating tools which are capable of collaboration, at first as receivers of direction, eventually perhaps as repositories of "knowledge", "skills", and strategic or tactical "insight" which would cast them as qualified "partners" in human activities.

The transition to a collaborative relationship with tools is a work in progress. Limited examples have appeared such as voice response telephone dialing, and speech directed surgical video cameras. As yet, no architecture, interface definition, or broadly useful base of enablers has emerged. The purposes of Personal Speech Assistant project include providing at least a test bed and an benchmark for low resource speech enabled tool direction.

The Personal Speech Assistant is a processing system occupying approximately three cubic inches, operating on two AAA batteries, and supplying a spoken command recognition engine, storage of speech for deferred dictation recognition, a user verification engine, a text to speech engine, personal area connection "engine" (IRDA 2), and a spoken user interface supervisor shell which manages the set of engines in order to structure a user interface from their capabilities.

I. Target Application

The Palm Pilot was selected as the first target for the Personal Speech Assistant because its strengths and weaknesses are a good complement to a speech system. The Palm Pilot is personal digital assistant intended by its designer to be "a window onto data". It provides very limited computation capabilities and is intended to be an "occasionally connected" device. It is sold with a set of Personal Information Management applications. In order to access, enter or modify data on the Palm Pilot, a user pushes buttons or enters Graffiti characters on a digitizer. User interactions with the system tend to be short and frequent. When connected, the Palm Pilot synchronizes its data bases with local and remote hosts and returns to its disconnected mode of operation. The Palm Pilot programming model is sufficiently open and straight forward that a large base of application software has been developed for it. Organizations such as Motorola and Delorme Mapping have used this PDA as window onto data for smart pagers, GPS mapping systems, and CDPD network connections. IBM re-brands the Palm Pilot as ThinkPad. Lotus supports synchronization of Palm Pilot data bases with corresponding Lotus data bases.

The Personal Speech Assistant Project
Liam Comerford, David Frank, Dominico Manildo, Steve Mastrianni, Jan Sedivy

The target application for the Personal Speech Assistant is a suite of "speech aware" PIM applications supporting deferred recognition (at hotsync time) for data entry and Text to Speech conversion for data access. This PIM suite will allow the user to control the PIM applications by voice command, be prompted by applications (for appointments for example), and to enter and access dictated data both before and after hotsync. The target set of PIM applications include a To Do List, Memo Pad, Calculator, Date Book, Phone Book, and eMail.

III User Interface

A speech enabler may provide any of three grades of command interface. The most concrete form associates speaking a command with activation of a control. If the current focus of the system is an application with a visual presentation, the command word could be title of the UI button the user wants to activate. Simple substitution of speech for manual activation of a control is useful as a development starting place, in cases where the user's hands are unavailable, and for noisy environments which mitigate against the use of high perplexity command vocabularies.

Simple substitution does not take advantage of the capacity of speech to address elements which are not present in the visual field or which cannot be easily represented visually. The second grade of interface is, thus, one which allows the user to avoid the structured set of dialogs presented by a GUI with shortcuts that provide the same information as is gathered by the dialog tree. For example, the sequence "File Open Look-in File-name Open" which involves a menu bar control, a drop down menu, and a dialog box, could be accessed with "Open File-name" and a spoken clarification transaction if the File-name is ambiguous. Similarly, large collections are hard to present, simultaneously visually and in detail. With speech, it is relatively easy to concatenate specifiers in order to arrive at manageable collections.

It should be understood that both of these grades of user interface require stereotyped behavior from the user. The user must utter the right word or a phrase in which the variable terms (like the File-name) are correctly positioned according to the command grammar. This is not to say that such interfaces cannot be done well. "Wildfire" is built at this level. The third form of user interface is one in which the meaning of the user's action is determined from a statistical model much as the word meant by a users uttered sounds are determined by the statistical model of speech in a speech recognition engine. Such a Natural Language Understanding engine accepts user utterances, maintains context information, and decodes the utterance and the context into a canonical representation which may then be used to determine, effectively, which buttons to press in what order.

In the near term, the means to support interfaces of the first and second kind are being implemented in the PSA. This will provide the infrastructure for future implementation of support for Natural Language interfaces.

Support for the first level of interface is provided in the form of dynamic vocabulary management. Each application state (corresponding to the current screen of controls in a GUI application) has an associated vocabulary. These vocabularies provide list of word, each

OIPE
SEP 11 2002
PATENT & TRADEMARK OFFICE

representing a control, and a command (and data) to be forwarded to the application in the event that the "control" word is recognized. Feedback and solicitation of additional information from the user can be made by the application through the use of an application specific prompt list or, more generally, through Text-to-Speech services in the PSA.

Support for the second level of interface is provided through the "Scripting" capability of the PSA. In addition to application specific vocabularies and prompts, an application may request that the PSA execute a "Phrase" from an application specific script. In general, a phrase can cause any engine to execute with pre or dynamically defined data and conditional execution of other phrases based on engine execution results. Dialog "repair" and complex "verbal" behavior in the PSA.

The set of capabilities, decoding speech, remembering speech, responding in speech, causing actions in attached devices can be used by the application developer to support a set of perceptions needed by the user in order to keep the dialog "on track" these perceptions include the impression that the system is listening, and the user is "understood" and that the system will "speak up" if these conditions aren't being met. Sets of vocabularies, prompts, and scripts for management of the PSA itself and for the dispatch of applications provide examples of this "confidence support".

The demands placed on application writers by the new and mixed modalities enabled by speech can also be explored in this environment.

II Hardware and Software structure of the PSA.

The PSA Shell comprises a dispatch component which manages the engine interfaces and an interpreter component which manages vocabularies, and prompts and responds to events generated by the Palm Pilot, the user, other communicating devices, or the engines, in accord with the actions described in the scripts. The interpreter drives the dispatcher, which manages resource and timing conflicts between engines.

In a PSA with personal area network support, detection of a new device which can be addressed can be accompanied by download from that device of command vocabularies, prompts, and scripts. In this way, as the user moves through the working, traveling and living environments, the PSA provides the Speech I/O capabilities to the local client devices. Among the immediate benefits this provides to developers of intelligent tools (Cars, TV sets, Houses, ...) is that the speech recognition system doesn't have to be duplicated in each appliance, users are identified and separated, and the user is only required to enroll on a single speech system with a single microphone.

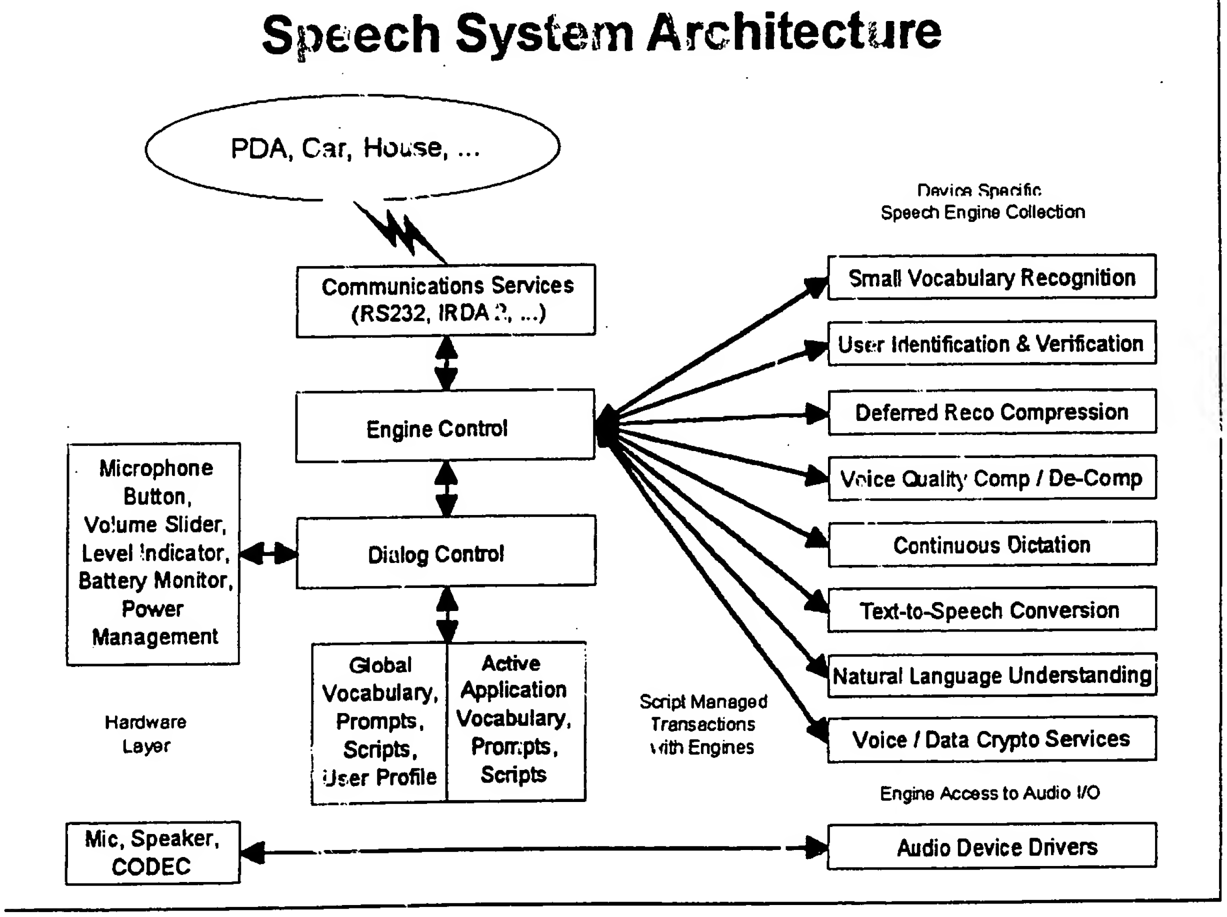
The engines and the PSA Shell are "low resource" speech technology. They are designed to execute under an embedded RTOS on a 40 MIP RISC processor.

III Physical Design

The Embedded Speech Interface Supervisor

The Supervisor is a software layer which provides specialized functions to an embedded application. These functions simplify the task of writing a speech aware application in much the same way that a BIOS simplifies access to the I/O hardware on a conventional PC. The Supervisor contains three functional blocks: Engine Control, Dialog Control, and the UI Data Store. The requisite engine initialization, housekeeping, and service request formatting are managed in the "Engine Control" component of the Supervisor. The Dialog manager, accepts events from the Engine Control component, User Interface hardware components (such as the microphone button), and the appliance or application, and uses data from the UI Data Store, provided by the application developer, to determine and launch the appropriate engine functions.

In the material below, the Supervisor is discussed in the context of the "PSA" or Personal Speech Assistant, a system comprising an embedded speech processor and a Palm Pilot.


Speech System Architecture
PDA, Car, House, ...
Device Specific Speech Engine Collection
Communications Services (RS232, IRDA ?, ...)
Engine Control
Dialog Control
Microphone Button, Volume Slider, Level Indicator, Battery Monitor, Power Management
Global Vocabulary, Prompts, Scripts, User Profile
Active Application Vocabulary, Prompts, Scripts
Hardware Layer
Script Managed Transactions with Engines
Small Vocabulary Recognition
User Identification & Verification
Deferred Reco Compression
Voice Quality Comp / De-Comp
Continuous Dictation
Text-to-Speech Conversion
Natural Language Understanding
Voice / Data Crypto Services
Engine Access to Audio I/O
Audio Device Drivers
Mic, Speaker, CODEC

The architecture of the PSA software system is shown in the diagram above. On the right side of the diagram, a set of human language service engines are shown connected to an "Engine Control" module. In addition, a Communications Services engine is shown above the Engine

Control module. Such engines are typically addressable by an API which offers controls that are typical of engines in general (Initialize, Start, Halt, Close, for example) and controls which are unique the individual engine function (Microphone-On for a speech decoder, Verify-user for a user ID and verify engine, decompress-and-play for a compression engine, etc. for example). Similarly, such engines typically provide status and data-containing messages which are general (ready, error, etc.) or unique (recognized-command, playback complete, etc.). A PSA may be embodied with as few as two engines (Command Recognition and Text to Speech) and still provide a useful, speech oriented interface. Further, any combination of engines (containing at least a Command Recognition engine and a sound output engine of some type) may be included, as is required by the user, the applications or appliances, the connection modes supported, and the capabilities of PSA hardware implementation.

The Engine Control module may be implemented as a receiver for engines configured as plugins or it may be written with knowledge of specific engines. The EC module performs initialization of the collection of engines and provides an abstract interface to their services. In addition, it provides notifications of events to the Dialog Control module. The DC module may have access to events generated directly on the PSA hardware (such as the microphone button and battery power level alarm) with or without the mediation of the EC module. Further the DC may have access to control of some PSA hardware components (such as status lamps) with or without the mediation of the EC module. These are implementation dependent choices.

The Dialog Control module accepts event notifications from the EC and other sources and search a collection of data files for "instructions" dealing with that event. It also maintains a scratch pad reflecting the current state information such as the microphone status, the current application identifier, and the activity status of the connected engines.

In order to illustrate typical operation of the Supervisor, transactions taking place during initialization are outlined below:

USER: Turns on power to the PSA
EC: sends "power-on OK" to DC
DC: searches message list for "power-on OK", follows instructions to search engine list, find hardware profiles, extract parameters, sends init engine messages to EC with pointers to appropriate parameter list.
EC: sends sequence of sets parameters with init commands to engines until list is exhausted, returns "init done" message to DC
DC: Searches message list for "init done", follows instructions to play "ready" prompt from prompt list by sending {tts play "I'm ready now"} to EC
EC: sends instructions to tts engine to play the string "I'm ready now".
USER: Hears "I'm ready now" and proceeds to use PSA

A second example may be seen when the user depresses the microphone button (while some text is being played by the TTS engine) and utters a command which changes the active application. In this example, the application is executing on an attached PDA such as a Palm Pilot, and the application has been written so that it can respond to messages received over its serial port.

DC: sees mic-button-down-event, looks at engine status list, sees TTS active, tells DC to halt TTS engine
EC: sends halt command to TTS, receives acknowledgment from TTS, sends "TTS Halted" message to DC
DC: changes tts status flag, sends {CommandReco Mic On} request to EC
EC: sends mic-on to Command Reco engine, receives mic-on status OK report form Command Engine, sends "CR On" message to DC
DC: changes cr status flag, turns on mic-on LED
User: speaks command to change application
EC: receives recognized word event, sends event notification with recognized word to DC
DC: Searches the vocabulary file associated with current application for the recognized word, fails to find word, searches application launch vocabulary for word, finds word, extracts the associated command sequence, sends {PalmPilot ChangeApp newapp} command to EC
EC: Packages the command sequence and sends it through the Communications Service engine to the Palm Pilot, sends "done" to DC
Palm Pilot: switches application, sends new-screen notification with identifier to PSA
EC: hears new-screen notification, sends "new target" event with identifier to DC
DC: updates local state data, searches list of user interface files for identifier related vocabularies, grammars, prompts, scripts, hardware setting profiles, sends EC sequence of re-initialization commands to tailor PSA behavior to the new application

The Supervisor allows the Speech Interface to cross application boundaries and to address application states which otherwise would have to be reached by tortuous GUI means. By writing the speech interface behavior as text containing vocabularies, prompts, grammers, etc., and providing a managed and unified interface to the set of speech service engines, the task of writing speech aware applications is reduced to little more than the conventional tasks associated with a non-speech aware application or appliance control program. Speech awareness is added by allowing messages received by serial port or other means to be seen by the application or control program event loop. Service requests from the application or appliance may be returned over the same channel. Single events and the commands which trigger them are specified to the PSA by text files which associate one or more vocabulary words with a command or command sequence. Similarly, spoken prompts can be specified in text files and decoded on application command. Other speech services are available through application specified scripts. TTS volume and voicing, microphone button behavior (push to talk, push to toggle, push to talk with activity time-out), and other hardware parameters, can be specified by the author or personalized by the user, in application specific profile files. In addition, the DC, finding that a screen or application has no associated user interface files, will execute a protocol requesting such files from a newly encountered application. Sets of default behavior and speech interface behavior files are provided in the base system.